Una Visión Arquitectónica de Sistema para Aplicaciones en Domótica

P. Lapine; *G.* Puschini; *E.* Wainerman; *A.* Crespo; *L.* Olsina
GIDIS - Dpto. de Computación, Facultad de Ingeniería, UNLPam
Calle 9 y 110 – (6360) Gral. Pico, La Pampa
TE +54 (0)2302 430497 Interno 6501
E-mail olsinal@ing.unlpam.edu.ar

Resumen: En el presente paper se ilustra una arquitectura de sistema para el control y monitoreo de módulos domóticos. Por una parte, la arquitectura de software se centra en un ambiente distribuido homogéneo pudiendo acceder a la funcionalidad de módulos domóticos, por medio de una Intranet o Internet. Por otra parte, se emplea una arquitectura distribuida CAN (*Controller Area Network*) en cuanto a la infraestructura para acceder a los dispositivos de sensado y actuación. Debido a que en trabajos anteriores se ha discutido la arquitectura de software, en este paper nos centraremos en las interfaces, capas y nodos funcionales para programar, controlar y monitorear un edificio así llamado inteligente. Se presenta consideraciones finales y futuros avances en el empleo de arquitecturas de software y hardware con potencial repercusión para el campo de la Domótica, entre otros dominios.

Palabras claves: Arquitecturas de Software y Hardware, CAN, RMI, Domótica.

1. Introducción.

En los últimos años los sitios Web han ido ganando en interacción y funcionalidad para pasar de ser no sólo una forma de presentar información y contenido sino también aplicaciones con soporte a complejidad de software tradicional. Así, muchas aplicaciones están siendo portadas a la Web, aprovechando las características de uniformidad, ubicuidad, seguridad y extensibilidad que este medio presenta.

En este contexto, el dominio de aplicaciones de Domótica se está viendo muy favorecida por el medio Web, entendiendo por Domótica –palabra surgida de doméstico e informática- como la disciplina que estudia el desarrollo de infraestructuras inteligentes en casas y edificios, como así también las tecnologías de información para soportarlas [7, 13].

Nuestra línea de investigación se centró en una primer etapa, en el diseño y construcción de una aplicación distribuida Web para el control y monitoreo de módulos domóticos, a saber: los subsistemas de Iluminación y de Cochera [5, 11]. En esta segunda etapa, desde el punto de vista de infraestructuras inteligentes, estamos integrando arquitecturas de software y hardware, centrada esta última en un sistema distribuido CAN [2, 8].

Desde el punto de vista arquitectural de sistema, se decidió por una arquitectura de software de tres capas en ambientes distribuidos homogéneos, con aspectos específicos para controlar y monitorear los dispositivos y eventos en tiempo real (esto es, *soft real-time*). Es de especial importancia el requerimiento de actualización del estado de los dispositivos en los clientes, ante los eventos ocurridos en el modelo físico, que es realizado por el servidor mediante el mecanismo denominado *callback* [5].

Por una parte, uno de los objetivos del proyecto fue la elección de una plataforma flexible y genérica para el desarrollo de aplicaciones centradas en la Web, evaluando las distintas arquitecturas e implementaciones posibles. Otra meta específica fue el crear componentes reusables basándonos para su construcción en distintos patrones arquitecturales y de diseño [1, 3]. Por otra parte, hemos desarrollado un prototipo de arquitectura distribuida CAN [4] (de uso en áreas de

control y automatización industrial), el cual favorece la reconfiguración conforme a los requerimientos de nuevos dispositivos de sensado y actuación, o ante fallas. CAN es un protocolo de comunicaciones basado en una arquitectura de bus para transferencia de mensajes en ambientes distribuidos. Entre sus fortalezas está el permitir una arquitectura multimaestro capaz de proveer características de respuesta en tiempo real (*hard real-time*) y tolerancia a fallas en la recepción de mensajes y mal funcionamiento de los nodos.

Debido a que en trabajos anteriores se ha discutido la arquitectura de software [1, 5, 11] en este paper nos centraremos en las interfaces, capas y nodos funcionales basados en CAN, para programar, controlar y monitorear un edificio inteligente. Además, discutiremos la integración de las arquitecturas de software y hardware respectivamente.

El presente trabajo se estructura de la siguiente manera: en la sección 2, se presenta un panorama del protocolo CAN, y se profundiza en aspectos de la arquitectura empleada en la sección 3.2. La integración de la arquitectura de sistema es analizada en la sección 3. En la sección 4, se discute detalles de la arquitectura CAN empleada para la implementación del módulo de Iluminación; y, finalmente se presentan las conclusiones y futuros avances.

2. Panorama del Protocolo CAN.

Como indicado previamente, CAN es un protocolo de comunicaciones basado en una arquitectura de bus para transferencia de mensajes en ambientes distribuidos. Fue originalmente concebido para aplicaciones en el área automotriz ([6], entre otros), pero rápidamente despertó una creciente atención en el área de control y automatización industrial donde los buses de campo *(field bus)* juegan un importante rol.

CAN considera una arquitectura multimaestro capaz de proveer características de respuesta en tiempo real y tolerancia a fallas en la recepción de mensajes y mal funcionamiento de los nodos. CAN está estructurado de acuerdo con el modelo OSI en una arquitectura colapsada de dos capas (esto es, capa física y capa de enlace de datos). Distintas opciones existen para la capa de aplicación, entre otras: CiA CAN Application Layer, CANOpen, SDS *(Smart Distributed System)*, DeviceNet y CAN Kingdom. La sección 2.1 trata sobre las propiedades fundamentales de la capa física CAN, en tanto que la sección 2.2 discute la capa de enlace de datos (tipos de servicios y control de acceso al medio).

2.1 Capa Física.

La capa física en CAN es responsable de la transferencia de bits entre los distintos nodos que componen la red. Define aspectos como, niveles de señal, codificación, sincronización y tiempos en que los bits se transfieren al bus.

En la especificación original de CAN [2], la capa física no fue definida, permitiendo diferentes opciones para la elección del medio y niveles eléctricos de transmisión. Las características de las señales eléctricas en el bus fueron establecidas más tarde por el estándar ISO 11898 [6]. La especificación CiA (CAN in AUTOMATION, http://www.can-cia.de), complementó las definiciones respecto al medio físico y conectores. Los nodos conectados al bus interpretan dos niveles lógicos denominados:

- ✓ *Dominante:* la tensión diferencial (CAN_H - CAN_L) es del orden de 2.0 V con CAN_H = 3.5V y CAN_L = 1.5V (nominales).
- ✓ *Recesivo:* la tensión diferencial (CAN_H - CAN_L) es del orden de 0V con CAN_H = CAN_L = 2.5V (nominales).

Esta denominación de los niveles lógicos responde al hecho de que un nivel dominante (0 lógico), sobrescribe un nivel recesivo (1 lógico), por lo que el bus desde el punto de vista lógico se comporta como una compuerta “y” cableada *(wired-and)*.

Cada bit es transmitido usando el código NRZ *(Non Return to Zero)*. Una característica del código NRZ es que la señal no provee flancos que pueden ser usados para resincronización si se transmite un gran número de bits con la misma polaridad. Por este motivo CAN usa bits de relleno *(bit-stuffing)* para asegurar la sincronización de todos los nodos en la red. La red opera en modo "quasi-estacionario". Es decir, por cada bit transmitido se da suficiente tiempo para que se estabilice el nivel de la señal en el bus antes que se realice el muestreo del mismo por todos los nodos en la red. A causa de estos requerimientos de estabilidad, la longitud máxima de la red depende de la tasa de transmisión que será empleada.

2.2 Capa de Enlace de Datos.

La capa de enlace de datos (DLL) está estandarizada por ISO 11898. Los servicios de la DLL en un nodo CAN son implementados por las subcapas control de enlace lógico (LLC) y control de acceso al medio (MAC) respectivamente. La subcapa de LLC provee las funciones de filtro de aceptación (son aceptados sólo los mensajes cuyos identificadores han sido previamente programados), notificación de sobrecarga y manejo del proceso de recuperación de error. La subcapa MAC es la responsable del empaquetamiento/desempaquetamiento de los datos, codificación de las tramas (*frames*), manejo de acceso al medio, detección de error, señalización de error y serialización/de-serialización de datos.

2.2.1. Estructura Básica de una Trama CAN. Cada mensaje intercambiado entre dos nodos en una red CAN, en principio está compuesto de dos campos tal como se observa en la figura 1. Un campo identificador (ID) y un campo de datos.

Fig. 1. *Trama CAN.*

El campo ID sirve a dos propósitos, a) para identificar el dato que está siendo transmitido en el bus, lo que significa que existe una relación directa y no ambigua entre dato e identificador; b) para permitir resolver el conflicto en caso de múltiple acceso. La unicidad de los identificadores permite arbitrar el acceso al bus entre los nodos que conforman una red CAN. El mecanismo de acceso al medio empleado por CAN es CSMA/DCR *(Carrier Sense Multi-Access with Deterministic Collision Resolution)*. Los nodos retrasan su transmisión cuando detectan la condición del bus como “ocupado”, lo cual ocurrirá mientras otro nodo transmite. Cuando se detecta la condición “bus ocioso“, un nodo puede comenzar a transmitir y el conflicto de múltiple acceso es resuelto a través de la comparación entre bits de los identificadores. Mientras no exista diferencia entre el nivel transmitido y el nivel sensado, no habrá finalizado la resolución del arbitraje.

Este mecanismo de arbitraje tiene las siguientes ventajas: 1) el de ser inherentemente no destructivo, debido a que el nodo que gana el acceso al bus continúa su transmisión sin retardo; 2) el de permitir priorizar un mensaje urgente en caso de múltiple acceso.

2.2.2. Servicios Implementados. El protocolo CAN provee dos servicios de comunicación tal como se ilustra en la Fig. 2, y se describe a continuación:

- ✓ *Servicio de Escritura*: una trama *(frame)* de datos se transmite desde un nodo denominado "productor", a uno o más nodos "consumidores". Este es el servicio de comunicación clásico de CAN.
- ✓ *Servicio de Lectura*: el servicio es iniciado por uno o más nodos consumidores, mediante una trama remota (*remote frame*) cuyo identificador corresponde a la variable que se desea leer. Esta trama tiene la particularidad de no poseer datos. El nodo productor de la variable responderá con la correspondiente trama de datos.

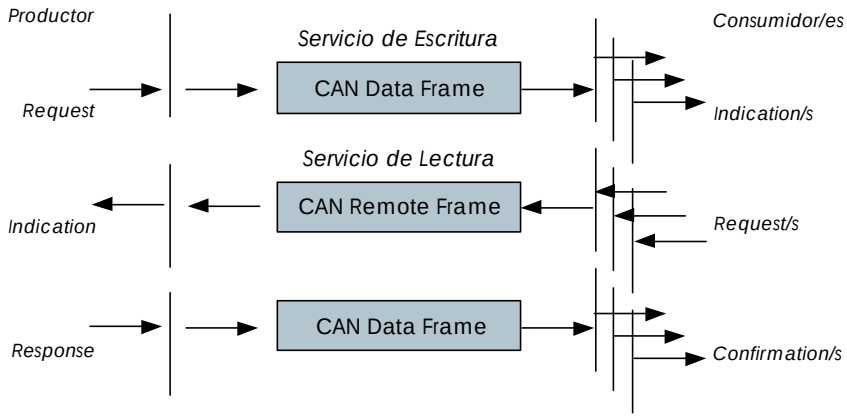

Fig. 2. *Servicios de comunicación del protocolo CAN*

3. Arquitectura de Sistema Distribuido aplicable a Domótica.

3.1. Arquitectura de Software.

Desde el punto de vista de la tecnología de información [13], la arquitectura de software utilizada se basa en un modelo de tres capas, a saber: la capa de *Presentación*, la de *Aplicación* y la de *Datos*. En la figura 3, se muestra un esquema general de la arquitectura de sistema propuesta, de software y hardware.

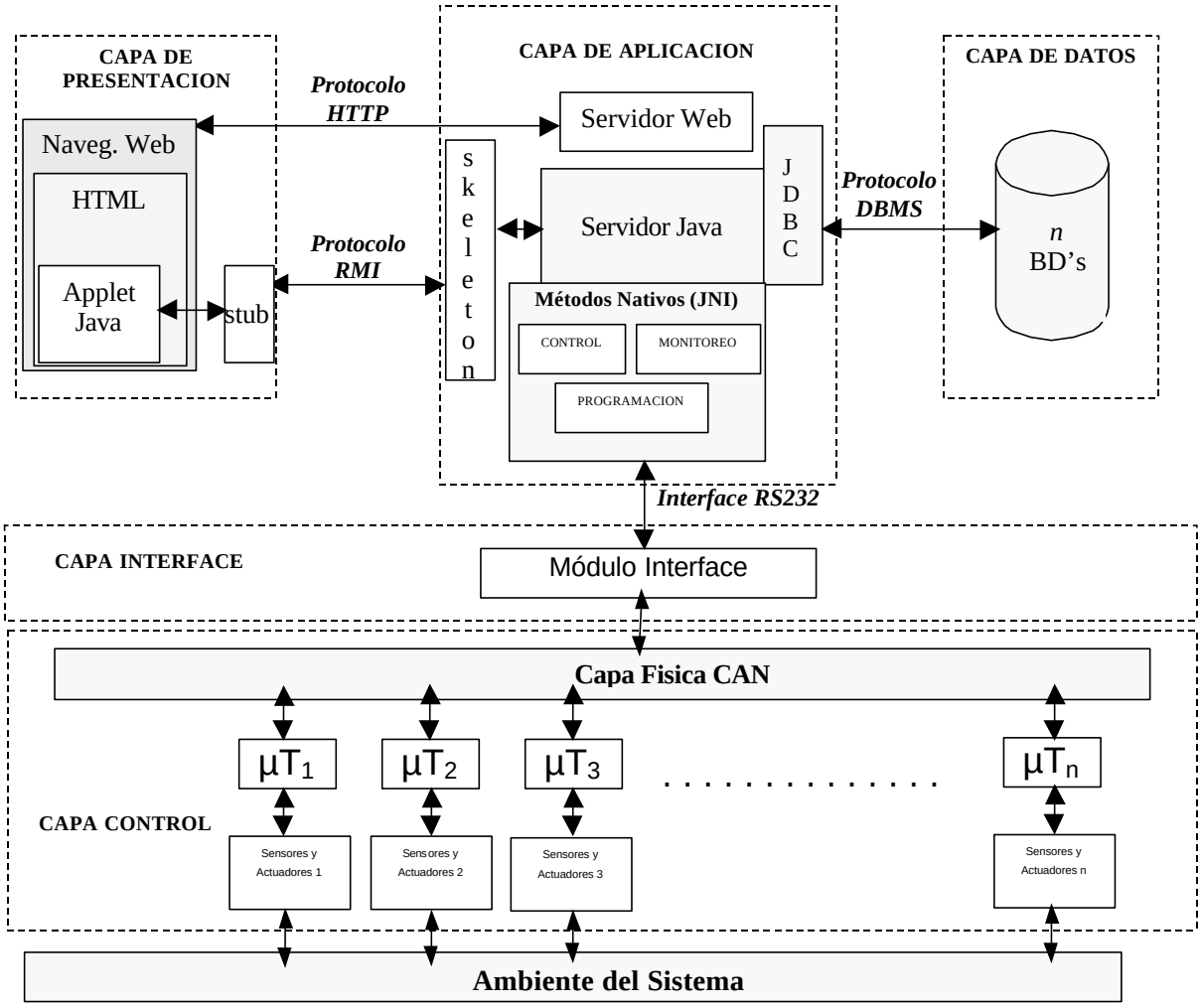

Fig. 3. *Una arquitectura de sistema propuesta para el control y monitoreo de funcionalidad Domótica.*

Respecto de la arquitectura de software, en la capa de *Presentación* se distribuye parte de la lógica de la aplicación (a través de applets, o mecanismos semejantes) y se consideran aspectos de interface de usuario. Los applets pueden ser ejecutados potencialmente en cualquier navegador pudiendo invocar tanto métodos locales como remotos. Para ejecutar métodos de objetos remotos, Java posee el mecanismo RMI *(Remote Method Invocation)* que permite la comunicación con el servidor de aplicaciones de una manera transparente para el usuario.

En la capa de *Aplicación*, se encuentra básicamente el servidor Web y un servidor de aplicaciones (o varios). Esta capa implementa gran parte de la lógica de la aplicación específica del dominio. Además se encarga de la manipulación de los datos (con la capa de datos) y de la comunicación y manejo de eventos con el módulo interface, que es el mecanismo que actúa de puente entre la computadora donde se halla el módulo servidor y las microterminales CAN, que controlan los dispositivos físicos. La comunicación entre el módulo servidor y el de interface se realiza a través de la interface estándar RS232 (como se discutirá en la próxima sección).

Para la manipulación de los datos, Java utiliza una interface de acceso a bases de datos llamada JDBC *(Java DataBase Connectivity)*. En la capa de Datos se encuentran todas las tablas a las que tendrá acceso la aplicación servidora.

La característica principal de los ambientes de software homogéneos es que, tanto el cliente como el servidor deben estar implementados en el mismo lenguaje para permitir su intercomunicación. El mecanismo RMI permite distribuir las tareas o procesos sobre otros objetos en la red. Es decir, no sólo se puede realizar paralelismo local (*threads*) sino también paralelismo global (en otras máquinas, potencialmente con otras plataformas operativas). Esta invocación a un objeto remoto es transparente al usuario, ya que éste no conoce en qué máquina reside físicamente dicho objeto, sino que los objetos remotos son vistos por el cliente como objetos locales. La transparencia es un atributo esencial de los sistemas distribuidos. Este trabajo fue ampliamente discutido en [5, 11].

En las siguientes secciones se analizan las capas de la arquitectura de hardware centradas en el protocolo CAN.

3.2. Arquitectura de Hardware centrada en CAN.

Desde el punto de vista de la arquitectura de hardware, el sistema consta de dos capas respectivamente, como se aprecia en la figura 3. La inferior se denomina *Capa de Control* o también se la suele llamar nivel de campo (ya que esta íntimamente relacionado con el medio). Su función es realizar el control distribuido de la aplicación. Esta capa se implementa mediante una red de control basada en CAN.

El nivel superior llamado *Capa de Interface*, se encarga de relacionar el modulo JNI del servidor de aplicaciones con la capa de control para entablar conexiones.

A continuación se describirán los bloques funcionales de las capas mencionadas.

3.2.1 Capa de Control. Esta formado por módulos CAN denominados microterminales.

3.2.1.1 Microterminales (mT). Habiendo definido las variables de muestreo y las variables de actuación, las μT son las encargadas de sensar datos, procesarlos mediante un algoritmo previamente programado, obtener el resultado requerido y actuar. Además permiten intercambiar información entre otras microterminales.

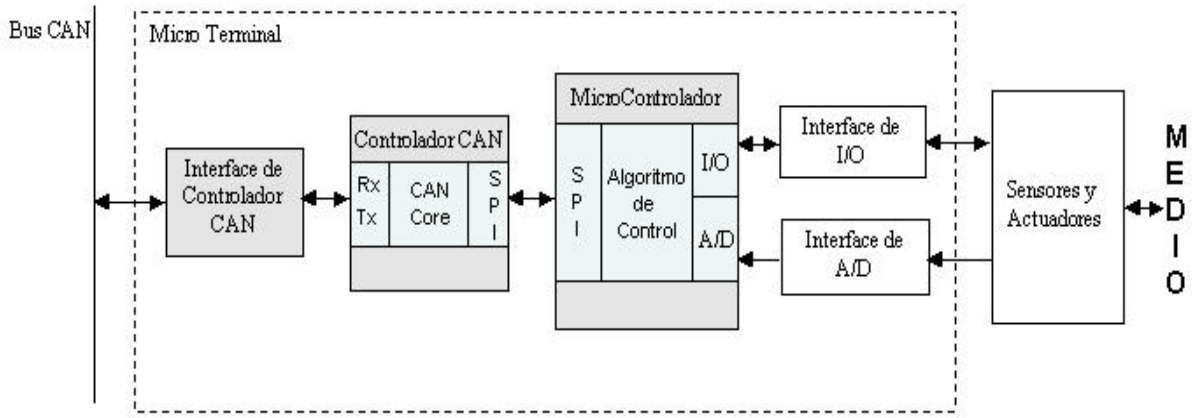

Fig. 4. *Diagrama de bloques de una microterminal*

En la Fig 4 se presenta un diagrama de bloques de una µT. Los bloques que se muestran sombreados están construidos por un único componente electrónico, cuya descripción se realiza a continuación:

Bloque Microcontrolador. Es la parte de la µT íntimamente relacionada con el control de la aplicación. Es el encargado de ejecutar el algoritmo de control. Para nuestro prototipo se utiliza el microcontrolador 16F877 de la empresa Microchip [9] que permite, entre otras funciones:

- ✓ Muestrear variables digitales y analógicas.
- ✓ Controlar variables de tiempo.
- ✓ Almacenar variables en un medio no volátil (memoria *flash*).
- ✓ Conectar periféricos mediante una interface serie, tanto sincrónica como asincrónica.
- ✓ Comunicar con otros microcontroladores vía interface SPI/I^2C (*Serial Peripheral Interface*/ *Inter-Integrated Circuit*).

Bloque Controlador CAN. Es un controlador *stand-alone* CAN 2.0b, y es el responsable de brindar la funcionalidad de red a las µT. Se encarga de realizar las siguientes tareas:

- ✓ Recepción y transmisión de mensajes.
- ✓ Empaquetamiento y desempaquetamiento de los mensajes según la trama CAN (ver sección 2.2.1).
- ✓ Filtrado del identificador del mensaje por filtros de aceptación, previamente fijados según la aplicación (sección 2.2).
- ✓ Chequeo de errores y validación del mensaje (sección 2.2).
- ✓ Clasificación del mensaje según el servicio requerido (sección 2.2.2).

Se utiliza el controlador CAN MCP2510 de la empresa Microchip. Las características que posee y son importantes a tener en cuenta para una determinada aplicación, son:

- ✓ Tres *buffers* de transmisión.
- ✓ Dos *buffers* de recepción.
- ✓ Seis filtros de aceptación.
- ✓ Interface SPI para la comunicación con otros microcontroladores.

Bloque Interface de Controlador CAN. Es un único componente electrónico denominado *transceiver*, que se ocupa de transformar los niveles lógicos estándares de tensión del controlador CAN a los niveles de tensión del bus (ver sección 2.1) y viceversa.

El *transceiver* utilizado es el PCA82C250 de la empresa Phillips [12], y sus principales características técnicas son:

- ✓ Máxima velocidad de transmisión de 1Mbit/seg.
- ✓ Una cantidad máxima de 110 nodos conectados al bus.

Bloque Interface de I/O. Como muestra la figura 4, es la interface de conexión entre el microntrolador y los dispositivos que se encargan de muestrear y actuar. Adecua los niveles de tensión y corriente de estos dispositivos a los niveles del microcontrolador. Otra función es la de proteger al microcontrolador de sobretensiones y cortocircuitos externos. Se selecciona según una aplicación determinada.

Bloque Interface de A/D. Dependiendo de los sensores que intervienen en el proceso, se utilizan amplificadores o atenuadores de señal, para convertir los rangos de tensión que nos entrega el sensor, a rangos de tensión estándar del microntrolador. En forma complementaria se puede usar algún filtro para minimizar los niveles de ruido en la señal. Estos componentes se seleccionan según una aplicación determinada.

3.2.1.2 Capa física. La capa física de conexión CAN ofrece el medio de transmisión, por el cual las µT intercambian información con otras µT y con el módulo interface.

Las µT y el módulo interface están conectados al bus CAN a través de dos cables denominados CAN_H y CAN_L (ver sección 2.1). En los extremos de estos, un par de resistores hacen de terminales (ver Fig. 5).

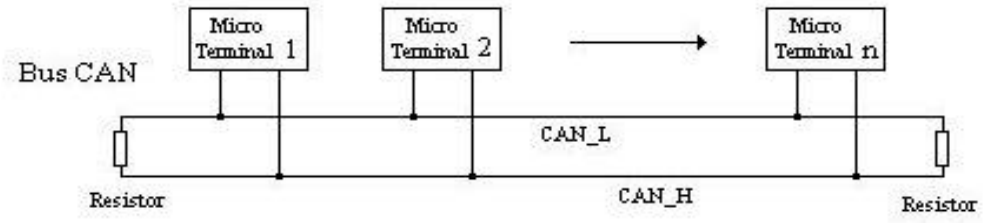

Fig. 5. *Esquema de la capa física CAN.*

El tipo de cable que se utiliza es del tipo par trenzado y el valor de los resistores es de 120 ohm. La tasa de transmisión de las µT depende de la máxima distancia de separación entre las mismas. La tabla 1 muestra la tasa de transmisión esperada que se obtendría en función de la máxima distancia de separación entre las µT. 6

Tabla 1. *Tasas de transmisión en función de la distancia máxima de separación entre las mT.*

Tasa [Kbit/seg]	Distancia [m]
1000	30
500	100
250	250
125	500
62.5	1000

3.2.2 Capa de Interface. Esta capa (según se aprecia en la figura 3), está formada por un único dispositivo llamado *Módulo Interface*. Su función es lograr una interface entre el servidor y las µT. En la figura 6 se muestra un diagrama de bloques de dicho módulo.

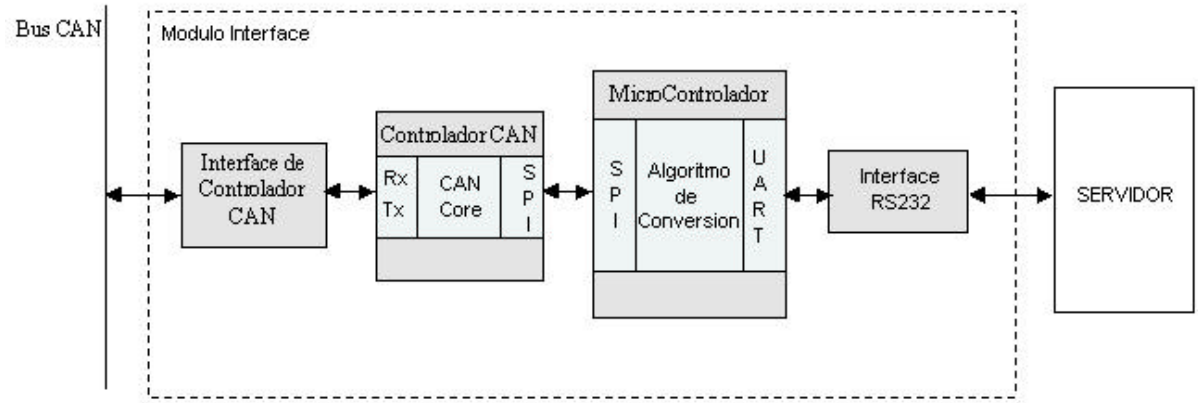

Fig. 6. *Diagrama de bloques del Módulo Interface.*

Este módulo es similar al de una µT, dado que utiliza el mismo microcontrolador, controlador CAN y la interface de controlador CAN (descriptos en la sección 3.2.1.1). Se diferencian en el algoritmo almacenado en el microcontrolador y en el uso de una interface serie asincrónica estándar RS232.

Para adaptar los niveles de tensión estándares de RS232 a los niveles de tensión del estándar del microcontrolador, se utiliza en nuestra experiencia un RS232 *Driver/Receiver* de la empresa National denominado DS14C232 [10].

4. Aplicación de la Arquitectura.

La experiencia de implementación de la arquitectura de sistema discutida, se realizó sobre un subsistema de iluminación. La funcionalidad apuntó al ahorro de energía y a la automatización de distintos aspectos de la iluminación de una maqueta edificio [11]. A seguir se describen algunos detalles de la aplicación de las capas inferiores de la arquitectura, a saber: Capas de Interface y de Control respectivamente, descriptas en la sección anterior.

4.1 Detalles del Sistema Implementado.

Primeramente se identificó el medio ambiente del sistema, definiendo las variables:

- ✓ *de muestreo*: esto es, nivel de luminosidad y estado de pulsadores (*on/off*).
- ✓ *de actuación:* para los circuitos de iluminación.

Al manejar poca cantidad de variables y a modo de una simple ejemplificación de la aplicación de esta arquitectura, se emplearon dos µT, denominadas: *Terminal Sensor de Luminosidad* y *Terminal Controlador de Luces*, pertenecientes a la Capa de Control; y un *Módulo Interface* perteneciente a la Capa de Interface.

El *Terminal Sensor de Luminosidad* tiene como función sensar variaciones de la luminosidad en el ambiente y enviar esta información a los nodos que la requieran.

El *Terminal Controlador de Iluminación* permite:

- ✓ Administrar en forma autónoma los circuitos de luces, adquiriendo información necesaria del estado de la luminosidad suministrada por la *Terminal Sensor de Luminosidad.*
- ✓ El control de los circuitos de luces por parte de los usuarios del sistema.
- ✓ Suministrar información a los usuarios relativa al estado de los circuitos de luces y sus respectivos cambios.

- ✓ Programar dinámicamente los parámetros de actuación.

El *Módulo Interface* es el puente entre las capas superiores y la capa de control (ver Fig. 3). Permite a la *Capa de Aplicación* mediante los *Métodos Nativos* (JNI) llevar a cabo las funciones de *Monitoreo* de mensajes y *Registración* de eventos en el bus CAN, y *Programación* en forma dinámica de los parámetros de actuación.

Dada la simplicidad de la aplicación, los nodos pertenecientes a la *Capa de Control* anteriormente mencionados, envían un sólo tipo de mensaje, es decir, utilizan un sólo identificador. Tomando en cuenta lo descripto en la sección 2.2.1, donde se explica la trama de un mensaje CAN, se ilustra a continuación la estructura de los mensajes.

El nodo *Terminal Sensor de Luminosidad* produce un mensaje caracterizado por su respectivo identificador y datos relativos al nivel de iluminación. La trama de este mensaje se muestra en la figura 7.

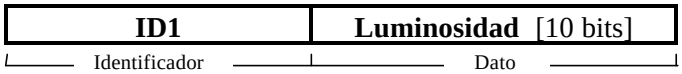

Fig.7. *Estructura de mensaje perteneciente a Terminal Sensor de Luminosidad.*

El nodo *Terminal Controlador de Iluminación* es productor del mensaje mostrado en la figura 8, reconocido por las entidades que participan del proceso de comunicación en la aplicación considerada, y caracterizado por el identificador ID2. El significado de los datos asociados al mensaje es subdividido en cinco partes, que se describen a continuación:

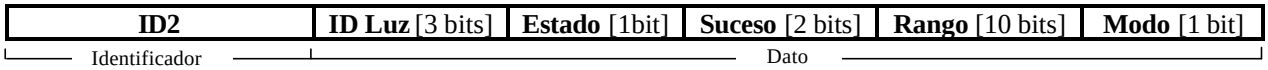

Fig. 8. *Estructura de mensaje perteneciente a terminal controlador de iluminación.*

- ✓ *ID Luz*: Identifica el circuito iluminación en cuestión. La cantidad máxima de circuitos de dispositivos de iluminación que se puede controlar en la aplicación planteada es 8 (campo de tres bits).
- ✓ *Estado*: Señala los dos estados posibles del circuito de iluminación identificado por el *ID Luz*, prendido (1) o apagado (0).
- ✓ *Suceso*: Informa qué suceso fijó el valor de la variable *Estado* perteneciente al circuito de iluminación identificada por el *ID Luz* . Existen tres tipos de sucesos, que para los bits:
  (00) - Forma automática: el valor de la variable *Estado* fue modificado por la luminiscencia del ambiente en cuestión.
  (01) - Forma manual local: el valor de la variable *Estado* fue modificada por el pulsador asociado a dicho circuito de luces.
  (10) - Forma manual externa: el valor de la variable *Estado* fue modificada por un usuario, que tiene acceso al servidor de la aplicación, a través de una Internet o Intranet, por medio de un navegador Web.
- ✓ *Rango*: Muestra el rango mínimo de luminiscencia perteneciente al circuito de iluminación identificado por el *ID Luz,* cuando la luminiscencia desciende por debajo de este nivel dicho circuito de iluminación se activará.
- ✓ *Modo*: Indica el modo de operación del circuito de iluminación identificado por el *ID Luz;* éste puede ser automático (1) donde el *Estado* es controlado por medio del *Rango* de funcionamiento; o manual (0) donde el *Estado* no depende del *Rango*, sólo depende de los usuarios, tanto locales (que acceden al control mediante pulsadores locales asociados a un

determinado circuito de iluminación), o externos (que acceden al control a través de un navegador).

El nodo *Módulo Interface* genera dos tipos de mensajes relacionados con los identificadores ID3 e ID4 respectivamente. Los mensajes transmitidos al *Terminal Controlador de Iluminación* tendrán rotulados el identificador ID3, y los mensajes dirigidos al *Terminal Sensor de Iluminación* son asociados al identificador ID4. La estructura de este mensaje se muestra en la figura 9.

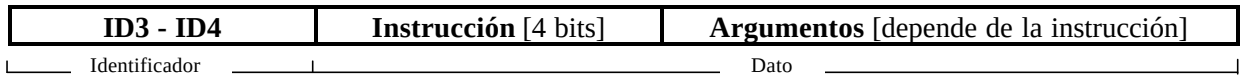

Fig. 9. *Estructura de mensaje perteneciente a Terminal Controlador de Iluminación*

Para esta aplicación, el *Dato* se dividió en dos campos, a saber:

- ✓ *Instrucción*: Define la operación que debe realizar una estación μT.
- ✓ *Argumento*: Información necesaria para ejecutar debidamente la instrucción.

En la tabla 2 se presenta el conjunto de instrucciones que son ejecutadas por las μT.

Tabla 2. *Conjunto de instrucciones pertenecientes a las mT respectivamente*.

Instrucción	**Descripción**	**Argumento**	
0000	Detiene el funcionamiento de la μT	-	
0001	Inicia el funcionamiento de la μT	-	
0010*	Programa rango mínimo de luminiscencia para el funcionamiento de un determinado circuito	*ID Luz* [3 bits]	*Rango* [10 bits]
0011*	Control manual de luces para los usuarios externos	*ID Luz* [3 bits]	*Estado* [1 bit]
0100*	Configura el modo de operación de un determinado circuito de luminiscencia	*ID Luz* [3 bits]	*Modo*[1 bit]
0101*	Peticiona información de un determinado circuito	*ID Luz* [3 bits]	
0110**	Peticiona estado de luminiscencia	-	

* Sólo es aplicable al Terminal Controladora de Luces.
** Sólo es aplicable al Terminal Sensor de Luz.

La asignación de prioridades a los mensajes hasta aquí considerados se muestran en la tabla 3.

Tabla 3. *Asignación de prioridades a mensajes.*

Identificador	**Prioridad**	**Procedencia del Mensaje**
ID4	ALTA	Modulo interface
ID3		
ID2	MEDIA	Terminal Controlador de Iluminación
ID1	BAJA	Terminal Sensor de Luminosidad

Para optimizar el flujo de la información, se programaron los filtros de aceptación de los respectivos nodos como se muestra la tabla 4, lo cual incide en una mejor eficiencia en la recepción de paquetes.

Tabla 4. *Programación de filtros*

Nodos	**Filtro de Aceptación**
Módulo interface	-
Terminal Controlador de Iluminación	ID1
Terminal Sensor de Luminiscencia	ID2

5. Conclusiones y Futuros Trabajos.

Domótica es un área que en estos últimos años ha comenzado a ser investigada y desarrollada, tanto desde el punto de vista de las infraestructuras inteligentes en casas y edificios como de las tecnologías de información para soportarlas. La experiencia presentada en este trabajo (y anteriores papers) de arquitectura de sistema integrada de software y hardware nos ha permitido realizar una aplicación distribuida para el control y monitoreo de un sistema domótico a través de una interface Web.

Dado que el acceso a las tecnologías Internet están cada día más difundidas y maduras, en especial el acceso a aplicaciones a través de la Web, y teniendo en cuenta que los dispositivos que permiten dicho acceso ya no son sólo las computadoras de escritorio, sino que hoy también se cuenta con dispositivos móviles como PDAs (*Personal Digital Assistant*), entre otros, vemos que este tipo de interfaces es muy útil para el desarrollo de aplicaciones domóticas.

La arquitectura de sistema analizada, además de los dispositivos Web del lado del cliente, necesita de al menos un servidor (computadora y plataforma operativa suficientemente robusta) a ser instalado en el edificio o casa, cuya responsabilidad es la de servir como puente entre los usuarios clientes y los dispositivos de sensado y actuación a controlar. Para este último fin, hemos seleccionado una red CAN, robusta y flexible, de uso cada vez más difundido en el área de automatización industrial.

Si bien la experiencia presentada, basada en arquitecturas de software distribuido en ambientes homogéneos, integradas a la arquitectura CAN, ha sido implementada en un proyecto a pequeña escala, nuestro desafío inmediato, es escalar este modelo y desarrollo a un proyecto de transferencia tecnológica para automatizar un edificio inteligente en la ciudad de Córdoba.

Agradecimientos. Esta investigación es soportada por el proyecto UNLPam-09/F014.

Referencias.

1. Ballari, T; Molina, H.; Wainerman, E.; Olsina, L.; (2001), *Uso de Patrones Arquitectónicos Web para el Diseño de una Aplicación Domótica*, Workshop de Investigadores en Ciencias de la Computación, San Luis, pp. 305-308.
2. Bosch, R.; CAN Literature; *http://www.bosch.de/de_e/productworld/k/products/prod/can/*
3. Conallen, J., (1999), *Building Web Applications with UML*, Addison-Wesley.
4. Crespo, A.; (2001), *Redes de Control: Análisis y Simulación del Protocolo CAN en un Escenario Real*. Tesis de Magister en Ingeniería Electrónica, Universidad Técnica Federico Santa María, Chile.
5. Echeverría, E.; T. Ballari; H. Molina; E. Wainerman; L. Olsina, (2000), *Arquitectura Centrada en la Web para el Control y Monitoreo de Funcionalidad Domótica*; Proceedings del VI Congreso Argentino de Ciencias de la Computación. (CACIC), Ushuaia, Arg.
6. ISO 11898, (1993), Road vehicles -- Interchange of digital information -- Controller area network (CAN) for high-speed communication
7. Jong Jin Kim, Jin Kim Jong (1998), *Intelligent Buildings*, Butterworth-Heinemann Publisher.
8. Lawrenz, W., (1997), CAN System Engineering: From Theory to Practical Applications, Springer Verlag; ISBN: 0387949399.
9. Microchip Inc., http://www.microchip.com
10. National Semiconductor Inc, http://www.national.com/
11. Olsina, L., Echeverría, E., Miguel, F., Giles, A. (2000), *Domotic Monitoring by using the Web*, Proceed. of AADECA 2000 (Asociación Argentina de Control Automático), Bs.As, pp 189-194.
12. Philips Inc., http://www.philips.com
13. Quinteiro González, J., Lamas Graziani, J.; Sandoval, J., (1999), *Sistemas de Control para Viviendas y Edificios: Domótica*, Ed. Paraninfo, Madrid. Spain.